NASA Technical Paper 3532

1995

Issues in ATM Support of High-Performance, Geographically Distributed Computing

Patrick W. Dowd
State University of New York at Buffalo
Buffalo, New York

Saragur M. Srinidhi and Eric D. Blade
Sterling Software
Cleveland, Ohio

Russell W. Claus
Lewis Research Center
Cleveland, Ohio

National Aeronautics and
Space Administration

Office of Management

Scientific and Technical
Information Program

1995

Issues in ATM Support of High-Performance, Geographically Distributed Computing

Patrick W. Dowd
State University of New York at Buffalo
Buffalo, New York 14260

Saragur M. Srinidhi and Eric D. Blade
Sterling Software
Cleveland, Ohio 44135

and

Russell W. Claus
National Aeronautics and Space Association
Lewis Research Center
Cleveland, Ohio 44135

Summary

This report experimentally assesses the effect of the underlying network in a cluster-based computing environment. The assessment is quantified by application-level benchmarking, process-level communication, and network file input/output. Two testbeds were considered—one small cluster of Sun workstations and another larger cluster composed of 32 high-end IBM RS/6000 platforms. The clusters had Ethernet, fiber distributed data interface (FDDI), Fibre Channel, and asynchronous transfer mode (ATM) network interface cards installed, providing the same processors and operating system for the entire suite of experiments. The primary goal of this report is to assess the suitability of an ATM-based, local-area network to support interprocess communication and remote file input/output systems for distributed computing.

Introduction

This report experimentally assesses the effect of the underlying network in a cluster-based computing environment. The assessment is quantified through application-level benchmarking and process-level communication.

Two testbeds were considered in this initial evaluation:

(1) A small cluster of Sun workstations (Sun Microsystems, Inc.)

(2) A larger cluster of 32 high-end RS/6000 (IBM Corporation) platforms—Lewis Advanced Cluster Environment (LACE) Cluster

The testbeds are located at the NASA Lewis Research Center in Cleveland, Ohio. The LACE cluster machines have Ethernet, fiber distributed data interface (FDDI), Fibre Channel, and asynchronous transfer mode (ATM) network interface cards installed, providing the same processors and operating system for the entire suite of experiments.

The primary goal of this report is to assess the suitability of an ATM-based network to support interprocess communication and remote file input/output (I/O) for distributed computing. The increasing importance of cluster computing is evidenced by the number of large corporations that have recently replaced their supercomputers with clusters. ATM cluster-based computing may provide supercomputing power without the excessive costs of supercomputers. A very valuable resource could be created even if a only a fraction of the idle central processing unit (CPU) cycles of workstations within an organization could be harnessed together. Many organizations already have a large quantity of high-end workstations that often sit idle. In addition, some organizations have geographically dispersed supercomputers and workstations, sometimes located in different cities. This report quantifies how the performance varies when different underlying networks are used to support such distributed computing.

ATM offers scalable performance and enables a stronger degree of integration between high-end and lower end processors. In combination with support for locality-exploiting load balance and process placement, ATM may make geographically distributed cluster computing possible. In this report, we begin to examine some of the basic issues necessary for implementing ATM as a large-scale computing resource.

A geographically distributed application would be partitioned into localities on the basis of communication patterns. There could be significant latency between localities, particularly if computing tasks were mapped to clusters in different cities, so the applications also would have to be partitionable. However, regardless of whether the objective was geographically distributed or local cluster-computing, support for low-latency communication within a cluster would be essential. Low latency is important at two levels in this context—the network and communication protocol. In addition, in cluster computing, a facility (such as a message-passing library) would be needed to support interprocess communication at the application level. This report examines the overhead incurred in supporting both levels and assesses the improvements possible with ATM.

We looked at both transmission control protocol (TCP) and user datagram protocol (UDP) as transport protocols to determine if a lower latency path was needed to the ATM adaptation layer via an application program interface (API). By varying the size and quantity of the data sent back and forth, we could collect the network's latency and throughput characteristics. From these data, we could determine the viability of the facilities.

This study determined the performance characteristics and associated overhead for ATM, particularly in the context of a high-speed network. The second area we looked at was our message-passing library—parallel virtual machine (PVM) (ref. 1). Other possible message-passing libraries include application parallel programming library (APPL) (ref. 2), message-passing interface (MPI) (refs. 3 and 4), and Express (ref. 5). Such libraries provide a distributed-memory environment to support cluster-based computing. Another environment is MNFS (ref. 6), which provides a shared memory view built on top of the network file system (NFS).

We first looked at basic performance measures to help characterize the library's PVM connection. Then the NASA NAS Parallel Benchmarks (NPB) (ref. 7) were used to determine the application-level performance implications of the processes that communicate through PVM. Message-passing library overhead, operating system issues (such as buffer moves), and protocol overhead were quantified in a cluster-based computing environment.

Because the volume of data generated in typical scientific applications can be very large, another important factor in cluster-based computing is distributed file I/O system management. Therefore, the problem of contention between interprocess communications and file I/O was examined, and the ability of the networks to support concurrent interprocess communication and network file system traffic was assessed (ref. 8). These studies will help us to assess the viability of certain networks, specifically ATM networks, in the cluster-based environment.

Experimental Platform

This section describes the network system configuration used for the experiments presented in the **Performance Evaluation** section.

Testbed Configuration

Two testbeds were used for the performance study. For the preliminary analysis, ATM was compared with Ethernet on a Sun-based testbed because ATM cards were not yet available for RS/6000's. The final assessment was done on an IBM RS/6000-based cluster.

Sun testbed.—This testbed consists of five SPARCstation series 10 workstations and one SPARCstation series 2 workstation (Sun Microsystems, Inc.) (fig. 1). All the workstations have ForeRunner SBA-200 ATM SBus adapter computer interface cards (FORE Systems) to perform segmentation and reassembly for AAL5 in hardware. All machines have a standard Ethernet interface card and are connected to the same subnet.

Two ForeRunner ASX-200 ATM switches (FORE Systems) (described in ref. 9) connect the workstations to the testbed. The switches are connected to each other by one TAXI line and two DS–3 lines. The DS–3 lines provide a throughput of 45 Mbps each, giving a total capacity between the switches of 190 Mbps. Three workstations are attached to each switch by TAXI lines that provide a peak capacity of 100 Mbps.

RS/6000 testbed.—This testbed (LACE) is a cluster of 32 high-end IBM RS/6000 workstations (fig. 2) configured to evaluate networking technologies. The LACE cluster has

(1) Two Ethernet interfaces per machine (One Ethernet is restricted to processors within the cluster to avoid contention caused by through traffic.)
(2) Sixteen workstations with FDDI interfaces
(3) Sixteen workstations with Fibre Channel interfaces
(4) Sixteen workstations interconnected via an ALLNODE switch (The ALLNODE switch, developed by IBM Corporation, is a

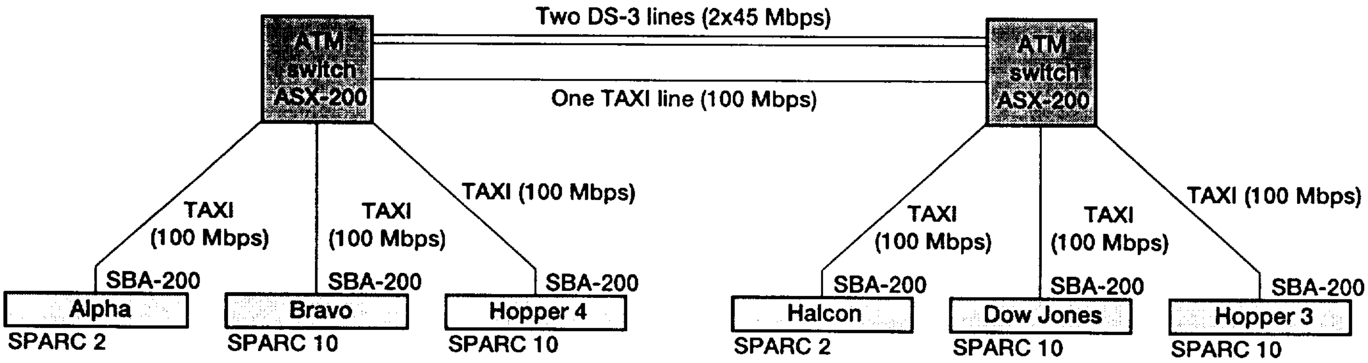

Figure 1.—Original ATM testbed based on Sun workstations.

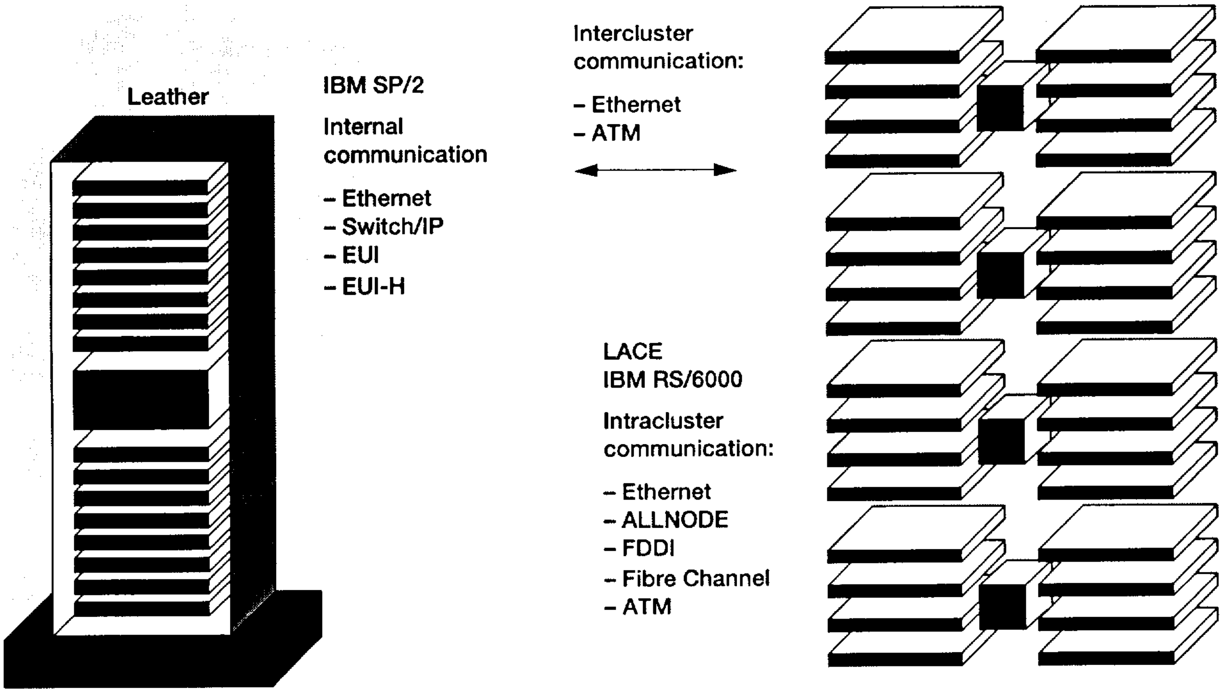

Figure 2.—ATM testbed on IBM RS/6000 workstations (LACE cluster) and the IBM SP/2 (Leather) multiprocessor.

multistage switch designed for clustered computing. This switch can support IP-level communication with low latency—on the order of 20 µs for small messages—through its proprietary, low-latency API. However, the processors must be physically close, such as on a raised floor.)

(5) Sixteen workstations with ATM network interface cards

All machines ran the same operating system (AIX 3.2.5, IBM Corporation), allowing for a uniform comparison of network technologies. In this testbed, only the networks varied.

We plan to create an environment that strongly integrates the IBM SP/2 (Leather) and the LACE cluster, both located at NASA Lewis (fig. 2). Although both systems can be used today, integrating these systems into a common environment will ease the job of parallel application developers.

Software Structure

This section describes the software structure of the system and discusses some limitations in the current architecture and our plans to address them.

Message-passing library.—PVM is used (and eventually MPI will be used) as the message-passing language for the following reasons:

(1) PVM is installed on an enormous number of machines.

(2) We can leverage the PVM ↔ ATM API development effort underway by the PVM development group at the Oak Ridge National Laboratory and the University of Tennessee.

(3) The IBM SP/2 provides PVMe, a version of PVM that avoids the TCP/IP (transmission control protocol/internet protocol) overhead and uses the high-speed switch for communication between processors within the SP/2.

ATM provides communication within the LACE cluster and between the LACE cluster and the SP/2. The SP/2 has two ATM interfaces. The idea is to provide a *best-path* communication strategy to support interoperability. The strength of this approach is a common environment that more closely integrates workstation clusters and supercomputers.

Software stack.—The software structure is illustrated in figure 3. In standard use, an application uses PVM or MPI as its message-passing library. A user can have multiple processes in a node that is using PVM. All the processes communicate via TCP/IP to a communication daemon within the node. The PVM daemon then communicates to other PVM daemons (in other processors) via UDP/IP.

A lower latency alternative is the PVM Direct Route feature that allows a process to directly communicate to another process with UDP/IP, thereby circumventing the routing daemon.

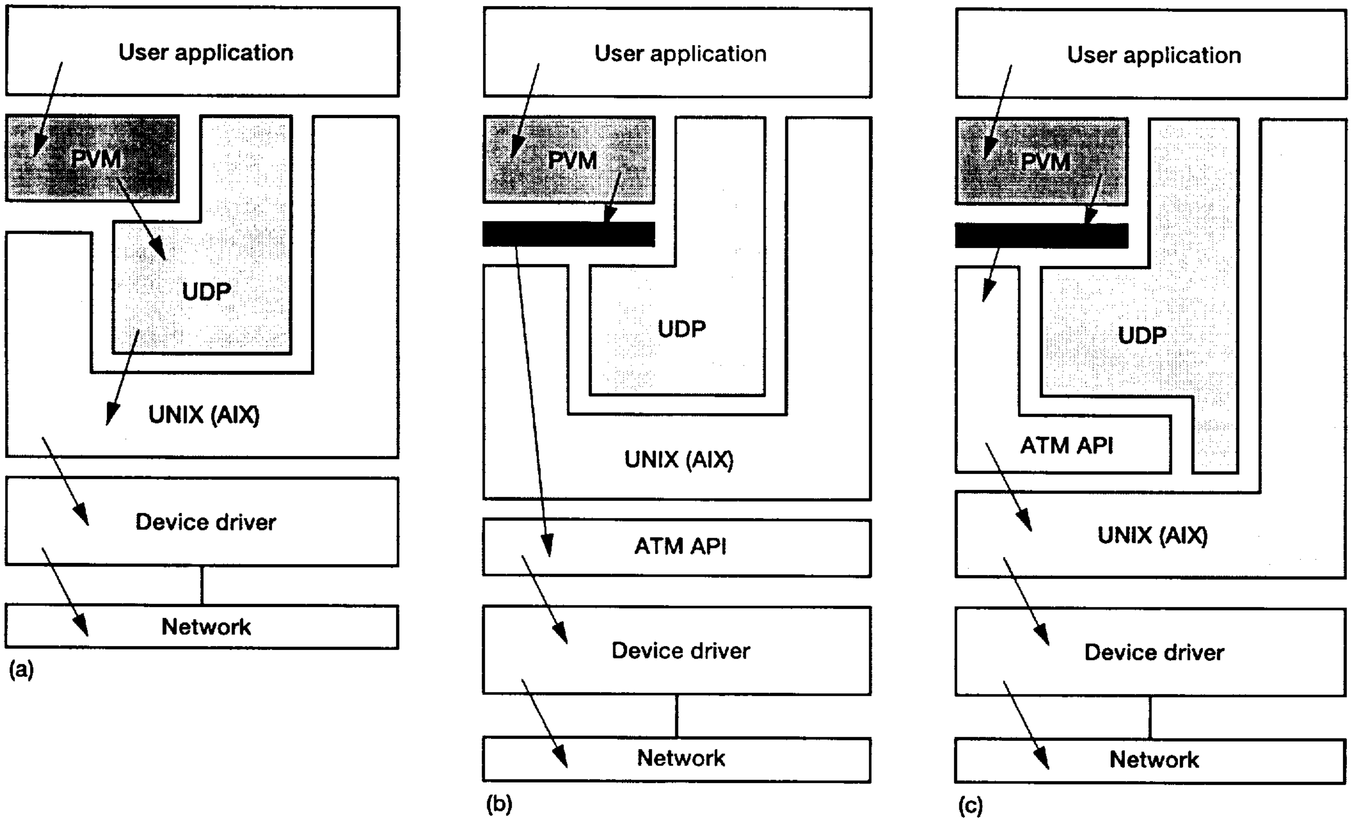

Figure 3.—Software structure supporting interprocess communication via a message-passing library such as PVM. (a) Current configuration. (b) Low-latency version with operating system removed. (c) Current API configuration.

In this model, the main contributors to latency are the delays due to system calls and to buffer moves between the different layers of the protocol stacks. This same strategy is used with Ethernet, FDDI, Fibre Channel, and ATM. IP is used by the ATM adaptation layer for name resolution; this also adds latency. The main advantage of the structure illustrated in figure 3(a) is that the details of the network are hidden from the user. However, the penalty in increased latency may be too high.

Figures 3(b) and (c) show an alternative where the message-passing library avoids the overhead of TCP and communicates directly with the ATM API. The two cases differ in the organization of the API. In the current configuration with FORE Systems ATM switches, the API is more closely depicted by figure 3(c) because system calls are incurred by the API with the operating system (and IP).

Currently, two of our projects that use FORE Systems ATM interfaces require us to examine and possibly modify the adapter and API code. Both projects use ATM as a high-speed data communication media in a distributed-processing environment. In particular, we are examining the usefulness of ATM in supporting cluster-based computing and also as a fabric to provide high-performance communication between supercomputers and high-performance workstation clusters (perhaps even when the processing elements are physically separated by large distances (e.g., in different cities). We want to determine if ATM will provide the level of integration and interoperability we require.

Geographically distributed computing.—Figure 4 illustrates the goal of the proposed effort: clusters of workstations fashioned into an integrated computing platform with seamlessly integrated, geographically distributed computing assets.

The desire to support this environment is based on the conjecture that supercomputers will remain expensive, in terms of price/performance, relative to workstations. Therefore, our goal is to use high-cost supercomputers more effectively and to more tightly integrate workstation clusters. This will provide users with "affordable" high-performance computing: the large number of preexisting workstations will be harnessed into a more effective computing environment, and supercomputers will be apportioned among many different groups. Figure 5 provides motivation for cluster-based computing. The graph illustrates the narrowing differences between microprocessor and supercomputer speeds: Microprocessor performance is rapidly gaining on the performance of single-processor and vector-based supercomputers. Figure 5 also shows how multiprocessor supercomputer systems scale, such as the SP/1, which is based on the IBM RS/6000 processor.

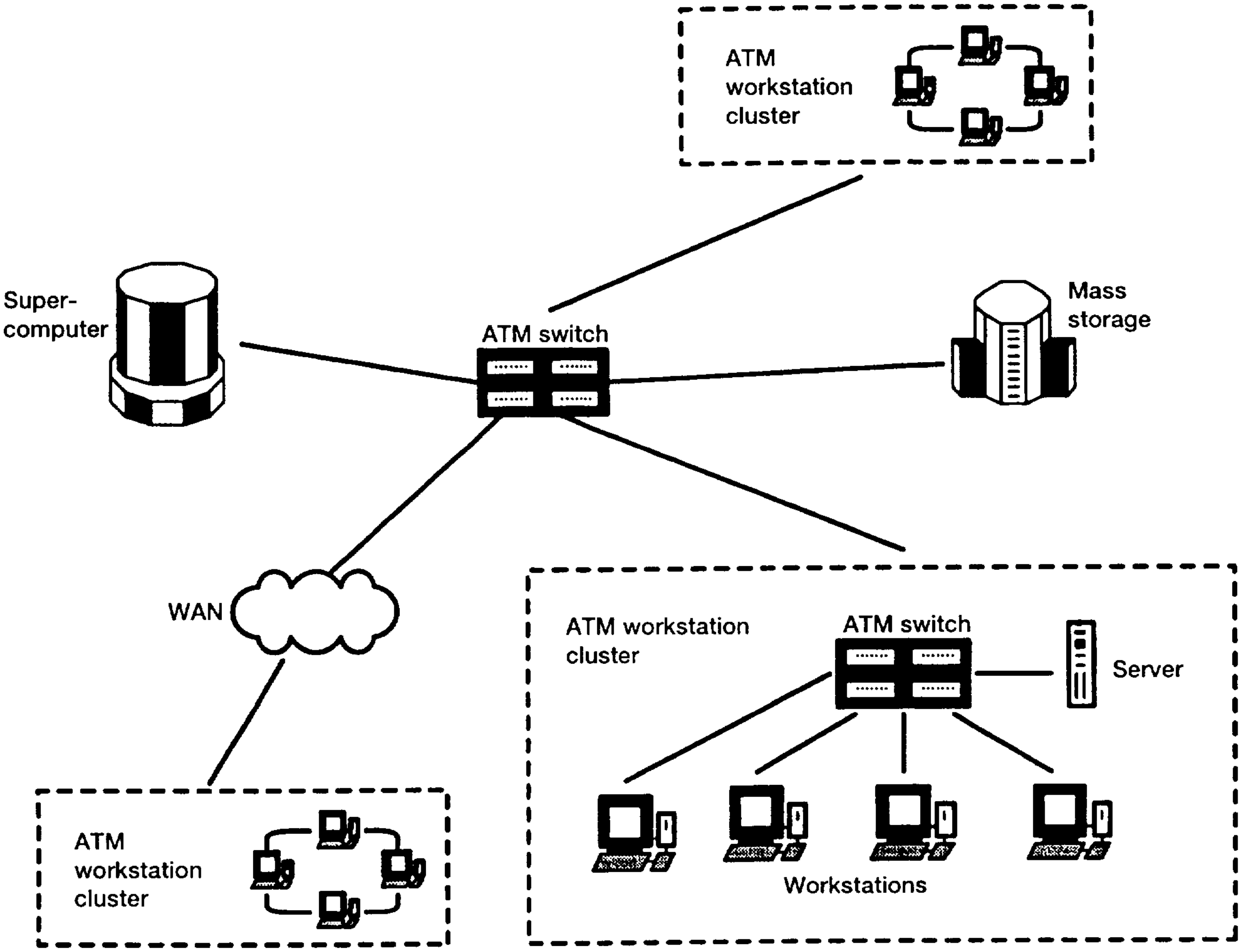

Figure 4.—Geographically distributed computing.

This approach deemphasizes the physical location of computing resources and allows people to more effectively use high-performance computing resources, such as supercomputers, while leveraging the existing base of high-end workstations in clusters to achieve affordable high-performance computing.

The target environment requires not just high throughput but also low-latency communication. When the connectivity patterns are not known a priori, low latency is necessary in a connectionless style of communication (as in permanent virtual circuit (PVC) or switched virtual circuit (SVC) with rapid bandwidth reallocation capability) (refs. 10 to 12). However, although low latency is essential within a local cluster, it cannot be achieved with geographically distributed computing because the propagation delay is a physical limit that cannot be circumvented.

Process placement: Suppose we had a parallel application that was written to execute on an Intel Hypercube (Intel Corporation). The connectivity patterns are well known, and we could map a virtual topology (refs. 13 to 15) to the ATM network so that the program could run without modification. However, this might not be the best way to take advantage of the strengths of ATM because ATM is far more flexible and could eliminate any intermediate routing by assigning virtual circuits to each pair of communicating processes. Some load balancing can be used in this context (refs. 16 to 18).

Low-latency support is required in a local environment. However, geographically distributed computing, enabled by ATM, will always have a higher latency because of the propagation delay. For example, the delay could be 50 to 250 μs between cities via land-line or satellite ATM. In this case, the application developer would need either to understand the network topology and write the application to suit it or to depend on a more sophisticated tool for process placement.

Overlap via distributed shared objects: A second strategy to desensitize the performance of the system to propagation delay is to overlap packets. Essentially, this strategy provides an aggressive prefetch support so that when information is required by a node it might not have to go all the way to the owner of that data—the data can be staged at a closer location.

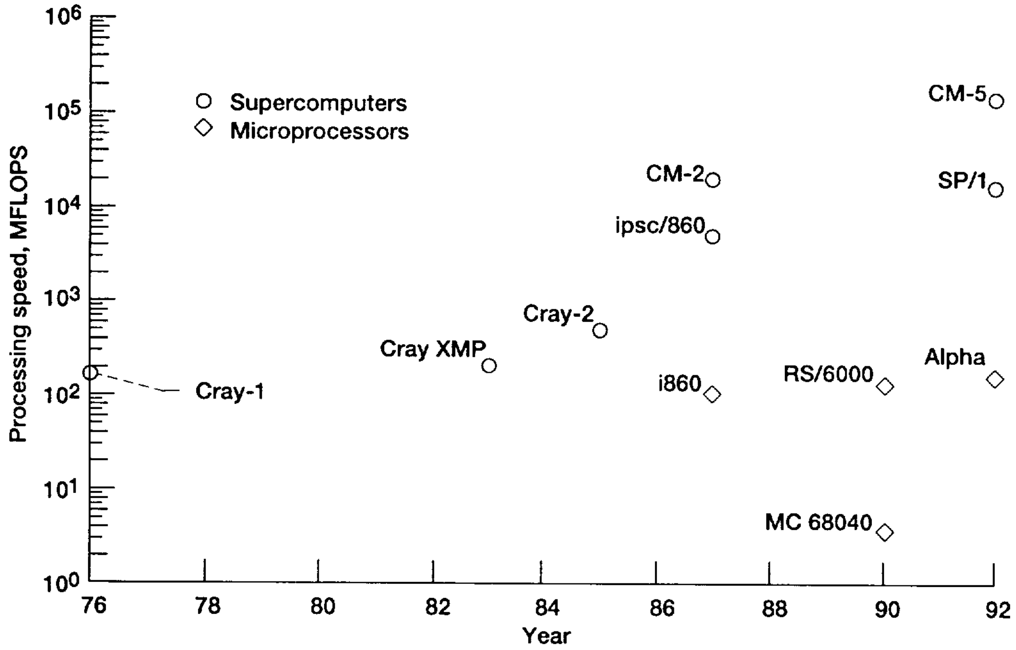

Figure 5.—Overview of relative performance of supercomputer; and high-end microprocessors.

This method implements a distributed shared memory—a functional, portable, and object-oriented shared memory. This distributed shared memory, which is a library of C routines that provide a weak-order coherence on a user-defined parallel virtual machine, can be installed on any computers running a Unix system without changing any kernel routines or adding any additional hardware. The experience we gained through two parallel application programs indicated that parallel programs can run successfully on a parallel virtual machine consisting of various types of computers under the shared memory environment, and that the program style is very similar to those sequential programs. The size of the parallel machine can be dynamically changed according to the requirements of the application programs and the availability of the computer resources. This facility is designed to operate at the message-passing library level. The major advantage of this is that it can be integrated into PVM, MPI, or APPL and does not require any modifications to the operating system kernel.

Asynchronous transfer mode (ATM) application programmer interface (API).—We are using ATM for its high speed. The API of the FORE Systems ATM interface was developed to avoid some of the TCP/IP overhead by having an application talk directly to a lower level of the system hierarchy. However, the API provided by FORE Systems (the manufacturer of the ATM switch) is designed to improve throughput over the ATM/IP path (refs. 19 and 20).

We need to reduce the overhead that diminishes the performance of ATM—not from a throughput point of view but with a primary objective of reducing the packet delay. With ATM, a packet (or segment) is chopped into many small cells. There is a signaling overhead in setting up the connection between the source and destination (called the cold-start latency), and there is a delay in sending a packet (including the segmentation and reassembly) when the virtual circuit already is in place (warm-start latency). We propose to dissect the API code provided by FORE Systems and to redesign it with support for low-latency communication as a primary objective.

Much of the difficulty in reducing latency results from using buffer copies and context switches and from invoking the operating system through system calls. It is our understanding that the FORE System's API was developed primarily to increase the maximum throughput. We want to examine the API to see if the latency can be reduced, perhaps with a slight tradeoff in throughput, while retaining the multiuser nature of the design. This is needed to exploit the PVM → ATM/API being developed by the PVM group at Oak Ridge National Laboratory.

Performance Evaluation

The performance evaluation was broken into three steps:

(1) Communication through the TCP/IP protocol suite to characterize interprocess communications

(2) Performance capabilities at the application interface level, with the TCP/IP stack still in place (at this level, the PVM message-passing library was used to examine communication between applications in a clustered environment)

(3) Performance capabilities when the protocol stack is avoided (access is made directly to the API of the ATM switch manufacturer)

In figures 6 to 10, the following cases are examined:

(1) Ethernet via both TCP and UDP

(2) FDDI via both TCP and UDP

(3) Fibre Channel via both TCP and UDP

(4) ATM via TCP, UDP, and the direct vendor provided API

Note that the throughput is described in terms of the maximum between two nodes. Depending on the protocol overhead, this could be close to the actual system maximum for shared media networks such as Ethernet and FDDI. However, ATM is a switched network, and the total system throughput could be much greater than this value, depending on the topology and the connectivity patterns.

Protocol Stack

The round-trip latency and the throughput of the networks through the TCP/IP and the UDP/IP protocol stacks were examined with a test program that sent a message to a host which then sent the packets back. Throughput and latency characteristics were collected for both directions. Utilizing both TCP and UDP, characteristics were gathered for all network connections. The size of the data transmitted by the test program was varied from 1 byte (B) to 8 kB for all four connections mentioned.

Much work has been done to optimize TCP to a high-speed environment. For example, references 21 and 22 show that the path length and latency can be significantly reduced when TCP is used as the transport layer protocol. However, we wanted to avoid using any unique or specific operating system modifications. Therefore, we restricted ourselves to off-the-shelf hardware and software environments. Also, no attempt was made to modify TCP or UDP beyond the standard distributions because our objective was to assess what is possible today, except for investigating access via the ATM API.

Connection assessment.—To separate transmission and setup characteristics, we performed tests with multiple-message and single-message transmissions. The objective was to isolate the initial connection setup time and identify it relative to the per frame and per message overhead. For example, figure 6 plots the latency of sending a 1-B message between two nodes using Ethernet, FDDI, Fibre Channel, and ATM as the underlying networks. The graph considers both UDP and TCP as the transport mechanisms. This illustrates the minimum delay possible with the networks including, for example, the connection time of TCP and the signaling overhead of ATM.

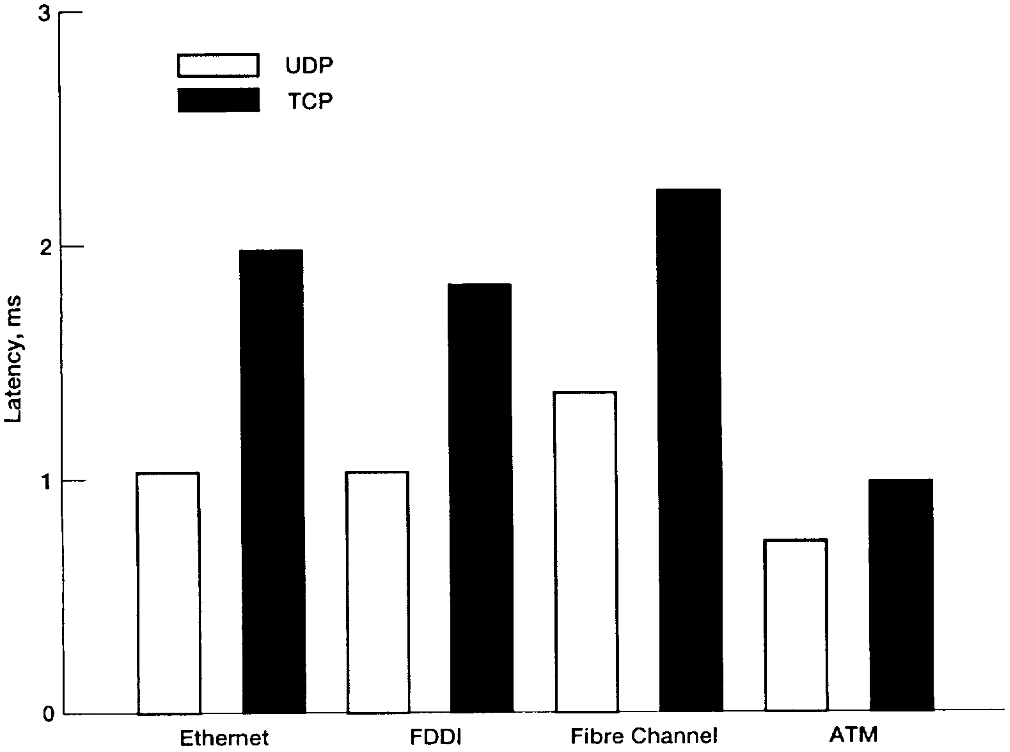

Figure 6.—Latency in sending a 1-byte (1-B) message, including startup overhead, for Ethernet, FDDI, Fibre Channel, and ATM. Each case considered both UDP and TCP as the transport protocol.

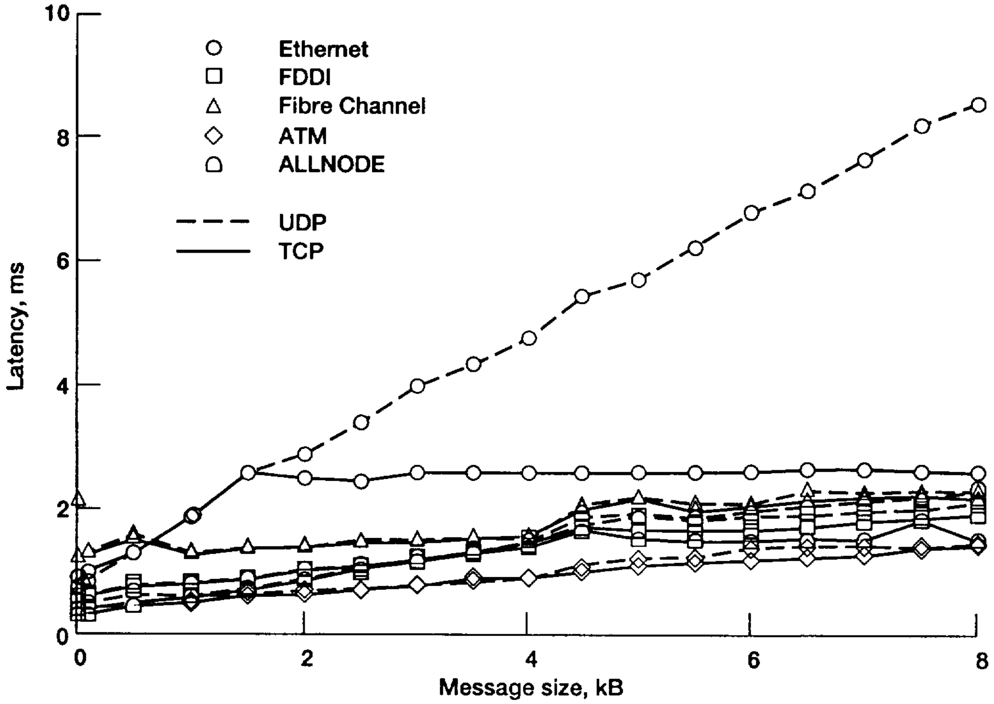

Figure 7.—Latency versus message size during startup, showing the costs of initialization for Ethernet, FDDI, Fibre Channel, and ATM. Each case considered both UDP and TCP as the transport protocol.

The next set of experiments apportioned the initial setup costs over many messages so that the steady-state behavior could be examined. For example, between 2000 and 3000 tests were run in the multiple message experiments, depending on what was needed to ensure statistical convergence.

The connection time is very apparent when there is only a single transmission, as in figures 6 to 8. For ATM, once a host is contacted via IP, a virtual circuit (VC) is maintained to that IP number. This VC is maintained after a message has been sent, and it remains in place governed by a least-used policy. For ATM this helps reduce overall latency by caching connections to a host. A connection is maintained until the ATM hardware address for the remote host is removed from the ATMARP table.

Figure 7 shows the latency versus message size for a single message. For this and the graphs to follow, latency is reported in milliseconds, throughput in megabits per second, and message size in kilobytes or bytes. Figure 7 shows that the signaling overhead is quickly allocated throughout the transmission of the multiple cells of the single message. The other networks also depend on their inherent maximum transmission unit (MTU). The cases examined are Ethernet, FDDI, Fibre Channel, and ATM. Each case uses UDP and TCP at the transport layer. The **Asynchronous Transfer Mode (ATM) Application Programmers Interface (API) and Parallel Virtual Machine (PVM)** section considers the case when TCP/IP or UDP/IP is not used and the ATM network is accessed directly via its vendor-provided API. These data, which were collected during startup, show the added costs of initialization. The latency is the time for a single message. It consists of multiple frames/cells when the message size exceeds the MTU size of the network.

Figure 8 shows the throughput versus message size. These are cold-start data, collected during startup. A single message was sent, so this includes the connection initialization (TCP, ATM, and logical link control (LLC)) overhead. Figures 9 and 10 give average statistics, showing how the overhead costs are apportioned over multiple messages. As expected, figure 8 illustrates the initial advantage of UDP in the avoidance of the transport-level connection.

For a very small message, the maximum throughput through UDP was about 0.006 for Ethernet, 0.012 for FDDI, 0.004 for Fibre Channel, 0.017 for ATM, and 0.025 Mbps for ALLNODE. With TCP the maximum throughput was 0.008 for Ethernet, 0.009 for FDDI, 0.006 for Fibre Channel, 0.0165 for ATM, and 0.021 for ALLNODE. For large single messages, the maximum throughput through UDP was about 8 Mbps for Ethernet, 31 Mbps for FDDI, 28 Mbps for Fibre Channel, 44 Mbps for ATM, and 28 Mbps for ALLNODE. The message size is varied above the MTU of the link/ATM level, showing how quickly each approach can recover the connection cost. Note that ATM via UDP can almost achieve the maximum throughput (which is constrained by the TAXI line speed of 100 Mbps), reaching a rate of 94 Mbps for very large messages.

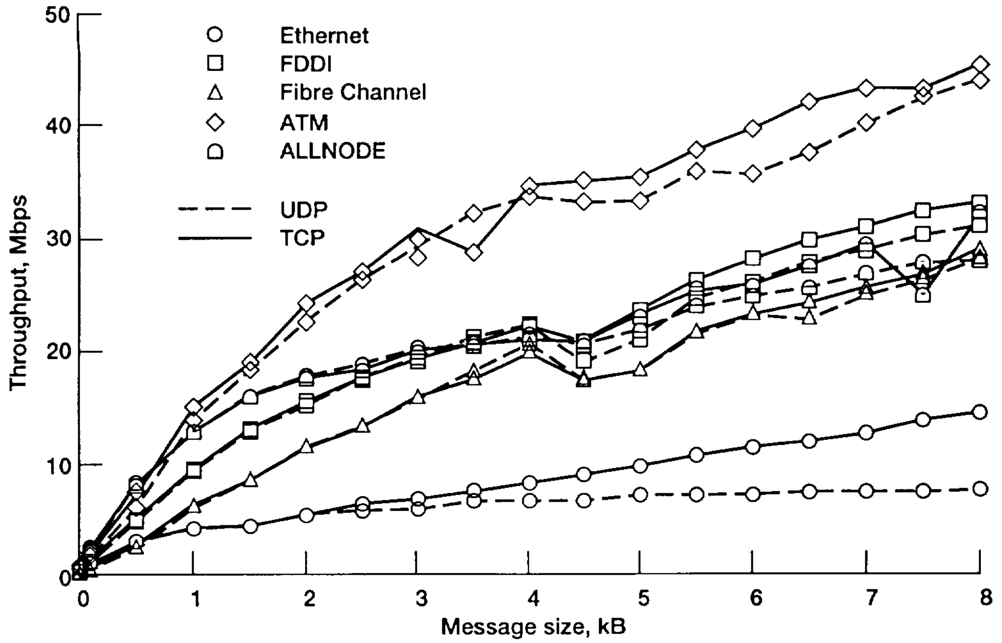

Figure 8.—Throughput versus message size during startup, showing the costs of initialization for Ethernet, FDDI, Fibre Channel, and ATM. Each case considered both UDP and TCP as the transport protocol.

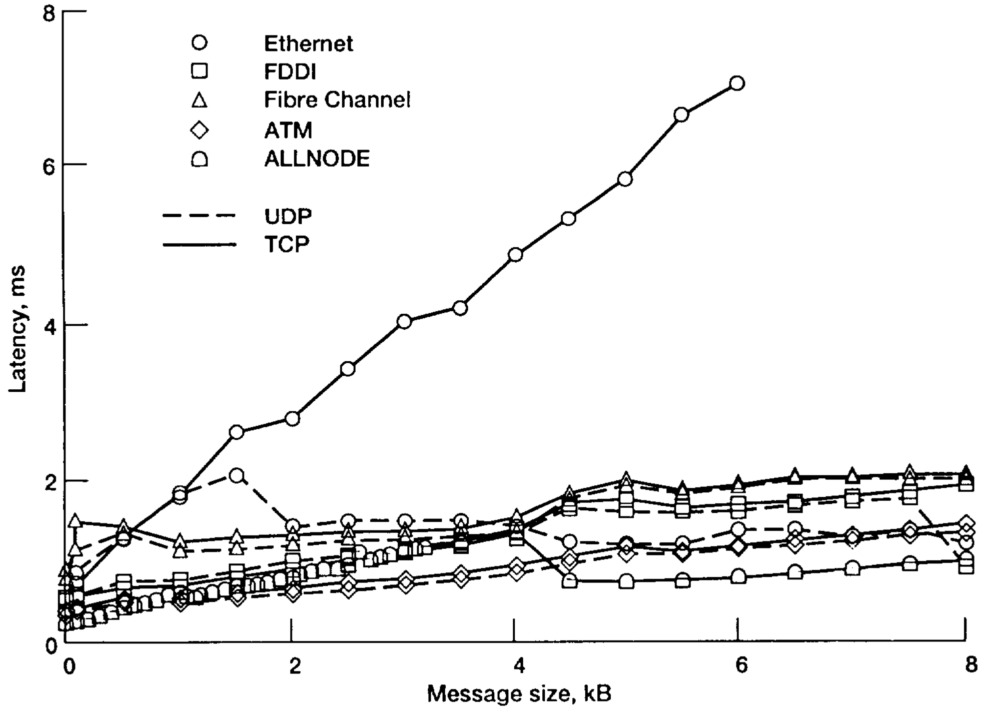

Figure 9.—Latency versus message size after startup, showing steady-state characteristics for Ethernet, FDDI, Fibre Channel, and ATM. Each case considered both UDP and TCP as the transport protocol.

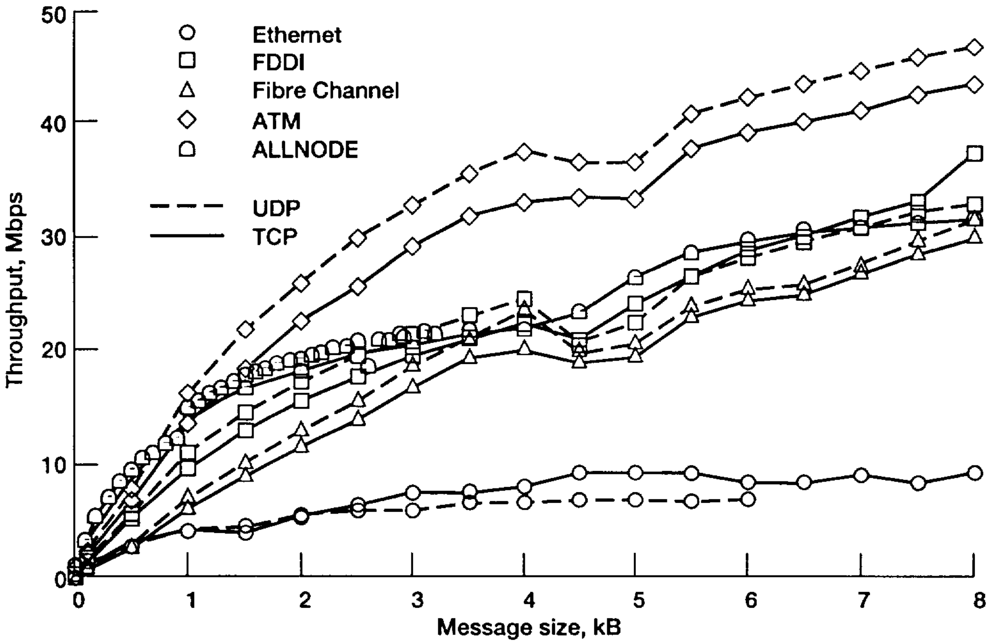

Figure 10.—Throughput versus message size after startup, showing steady-state characteristics for Ethernet, FDDI, Fibre Channel, and ATM. Each case considered both UDP and TCP as the transport protocol.

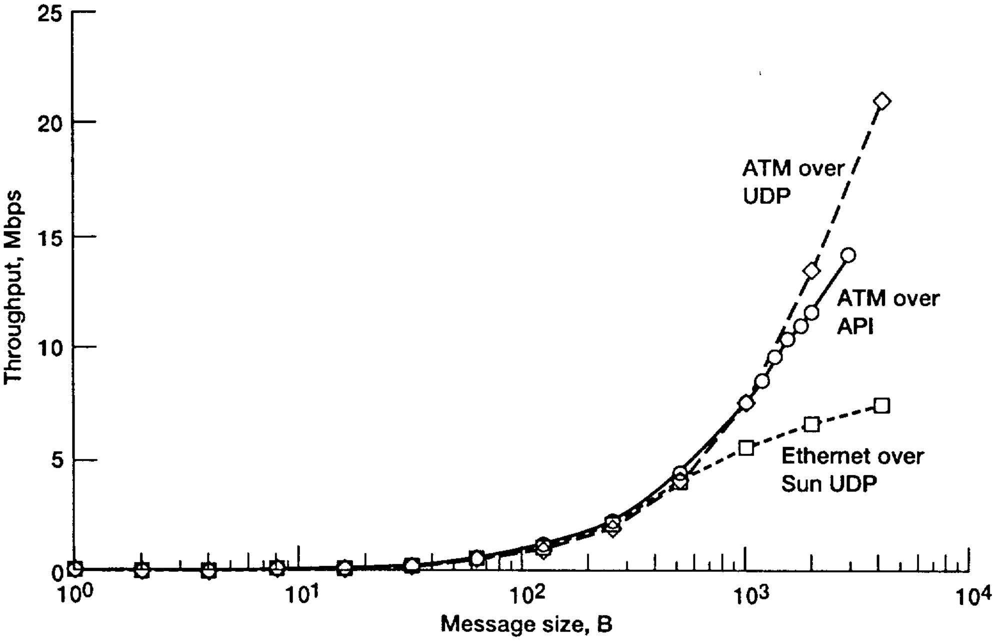

Figure 11.—Comparison of throughput versus message size for API with ATM over UDP stack and Ethernet over UDP.

Allocated connection overhead.—Figures 9 and 10 assess the steady-state performance of the networks under consideration when the initial startup costs have been apportioned over many messages. Figure 9 shows the latency versus message size, and figure 10 shows the throughput versus message size. The same network and transport level cases were considered as in the ***Connection assessment*** section. These data were collected after startup and show the steady-state characteristics. Figures 7 and 8 consider the latency and throughput, respectively, for startup transmission. Startup costs of the connection can be seen directly as the distance between the two traces. In this example, the latency for cold transmission is compared with the communication latency with a preexisting connection. The ATM tests were repeated with an artificially created load placed across both switches.

In an effort to improve the latency characteristics, the message-passing library was modified to directly access the ATM API that provides a direct link to AAL 5. The objective was to determine the magnitude of the performance improvement compared with that for ATM over TCP/IP. The PVM library was rewritten to communicate directly to the ATM adaptation layers.

However, figure 11 and 14 show that there is little benefit of direct API access in the current implementation of the ATM network interface adapters and switches: The throughput when using ATM through UDP is actually superior to using the ATM API. The current implementation of the API uses STREAMS to communicate with the device driver, causing a context switch, whereas the UDP protocol is implemented entirely within the kernel. The TCP/IP implementation consists of kernel-level functions, whereas the API is a process-level library. We are working with the switch manufacturer to address this concern by improving the API for low-latency access (ref. 20).

Asynchronous Transfer Mode (ATM) Application Programmer Interface (API) and Parallel Virtual Machine (PVM)

The PVM message-passing library uses the TCP/IP protocol stack for process interconnection. Specifically, PVM uses UDP for connection between its processor daemons. The throughput and latency of this implementation are examined in this section. The characteristics for the standard UDP connection, for both Ethernet and ATM, are considered.

For various transmitted packet sizes, figures 12(a) and (b) show latency and throughput for the systems using PVM. The graphs show that, although the message-passing library adds additional overhead and the percent reduction or increase in latency or throughput is somewhat degraded, the performance advantage of high-speed networks, such as ATM, is maintained.

The advantage of enabling PVM to communicate directly to the ATM is considered in figure 13(a)—round-trip latency versus message size. Three cases were considered: PVM communicating over Ethernet, PVM communicating over ATM via IP, and PVM communicating over ATM via the ATM API without TCP or UDP. This graph shows that there is only a slight improvement in capacity. However, if the scale of the graph is reduced as in figure 13(b), we see that latency increases significantly when the API is used. In fact, this graph shows that the latency in using the PVM over ATM via the API port is almost an order of magnitude greater than in using the usual PVM over ATM via UDP/IP and PVM over Ethernet. According to the designers, the FORE Systems API was not devised to achieve low latency, but to improve throughput. Another port to the ATM API, which uses the PVM Direct Route feature and was developed by the designers of PVM, also is shown in figure 13. This port (PvmRouteAal) has more latency than the nominal method, but with a less severe increase. Figure 14 shows the throughput resulting from the different methods. Both versions of the PVM port to the ATM API lower throughput in relation to the usual path through UDP. However, this is only true with a message size of up to about 1000 bytes. For large messages, both ports achieve a larger throughput.

In an effort to improve the latency characteristics, we modified the message-passing library to directly access the API that provides a direct link to AAL 5. The objective was to determine how much performance improved in comparison with ATM over TCP/IP. The PVM library was rewritten to communicate directly to the ATM adaptation layers.

Application Level Communication

The NAS Parallel Benchmarks (NPB) were developed at NASA Ames Research Center to study the performance of parallel supercomputers (ref. 7). The NPB are a set of eight benchmark problems, each of which focuses on some important aspect of highly parallel supercomputing for aerophysics applications. The eight problems consist of five "kernels" and three "simulated computational fluid dynamics (CFD) applications."

The benchmark problems are specified in a *pencil and paper* fashion. Refer to reference 7 for a comparison of the benchmark results with most supercomputer systems. The complete details of the problems are given in this reference, and except for a few restrictions, benchmark users are, for the most part, free to select the language constructs and implementation techniques best suited for a particular system. The NAS benchmarks, BT, LU, and SP, which were run on all the configurations mentioned previously, are intended to accurately represent the principal computational and data movement requirements of modern CFD applications:

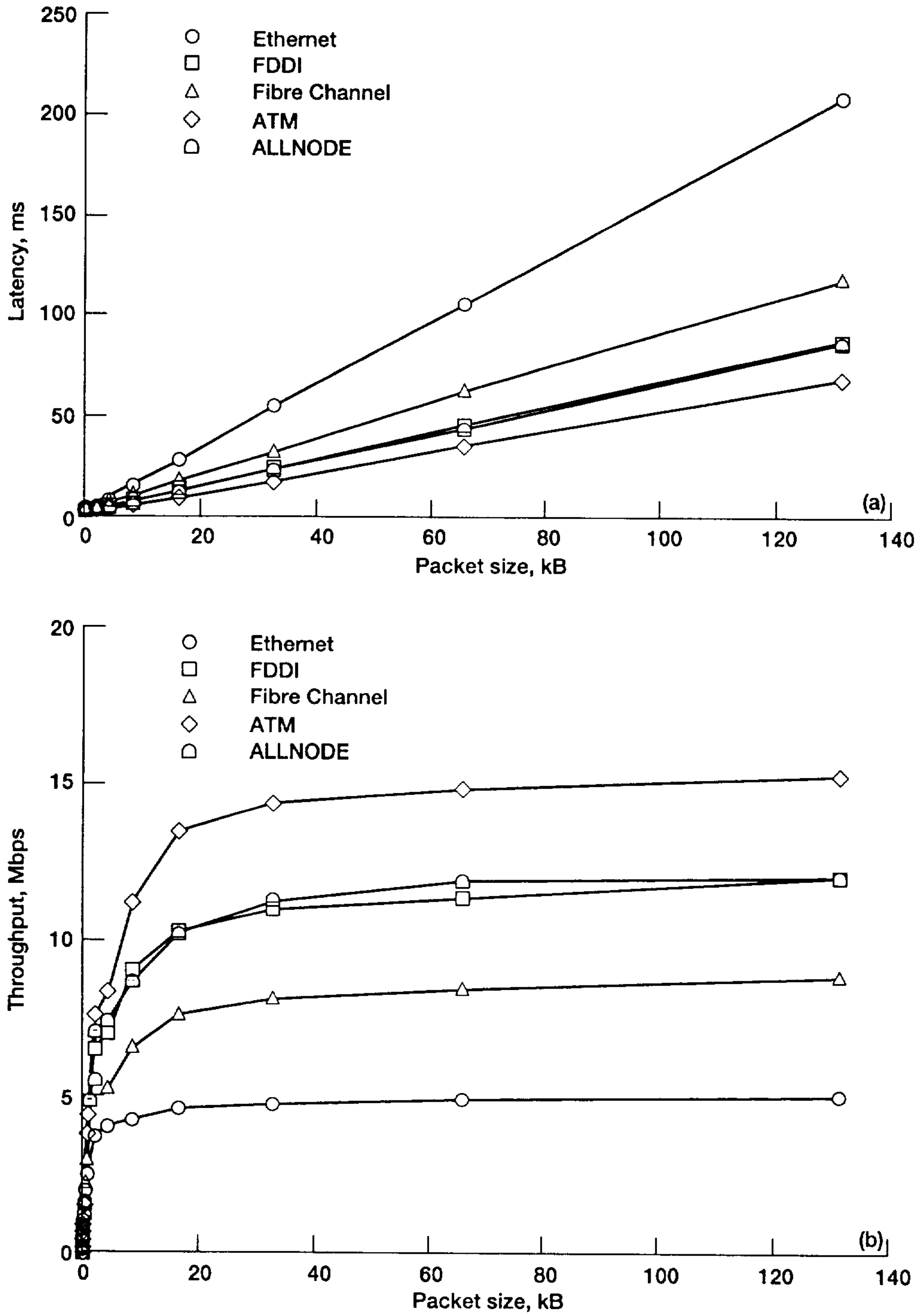

Figure 12.—Performance achievable through a message-passing library (PVM). (a) Latency versus message size. (b) Throughput versus message size.

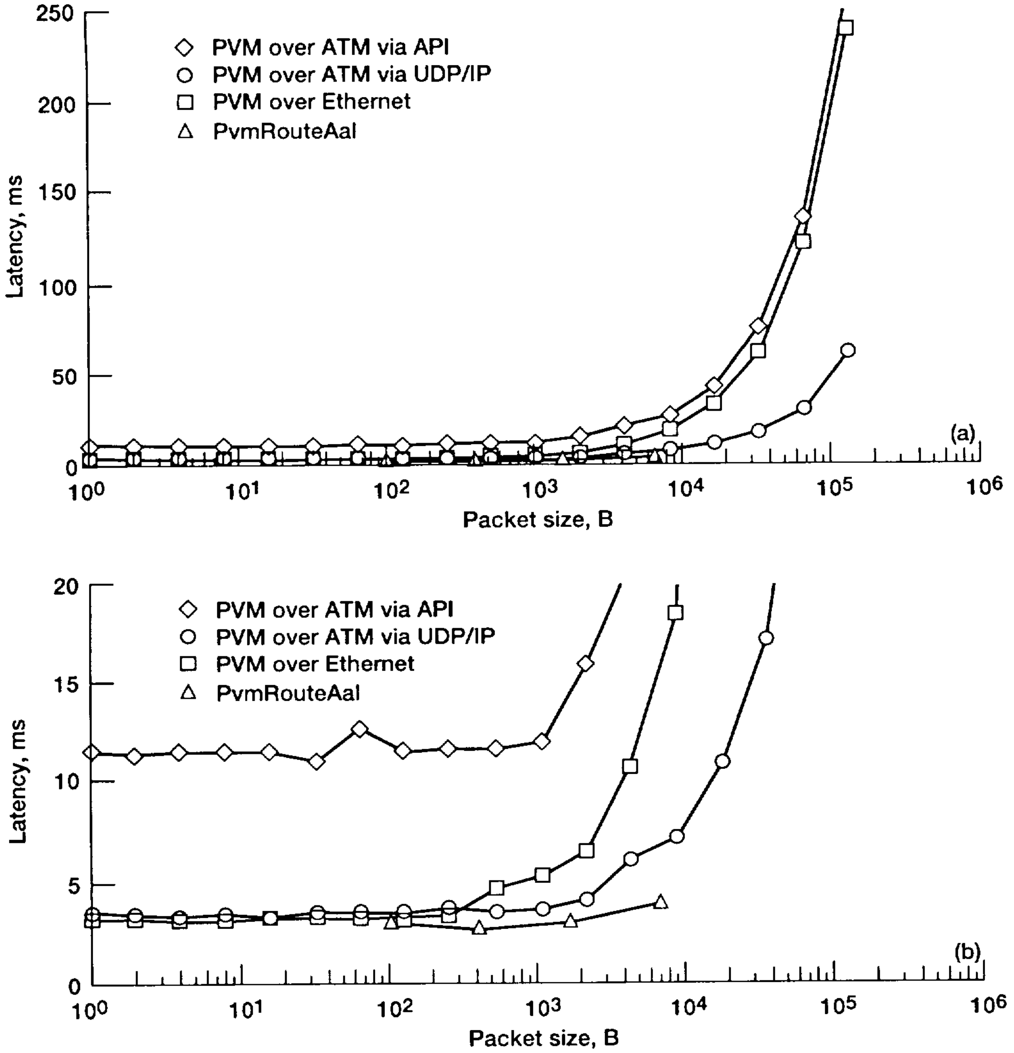

Figure 13.—Round-trip latency versus message size with PVM showing the effect of having PVM communicate directly to the ATM API. (a) Large-scale comparison. (b) More detailed comparison.

LU This application is called the lower-upper diagonal (LU) benchmark. It does not perform an LU factorization but instead employs a symmetric successive overrelaxation numerical scheme to solve a regular-to-sparse 5-by-5 block lower and upper triangular system. This problem represents the computations associated with a newer class of implicit CFD algorithms, typified at NASA Ames by the code INS3D–LU.

SP This simulated CFD application is called the scalar pentadiagonal (SP) benchmark. Multiple independent systems of nondiagonally dominant, scalar pentadiagonal equations are solved in this benchmark.

BT This simulated CFD application is called the block tridiagonal (BT) benchmark. Multiple independent systems of nondiagonally dominant, block tridiagonal equations with a 5-by-5 block size are solved in this benchmark.

Figure 15 shows these three benchmarks for different networks on the RS/6000 test bed.

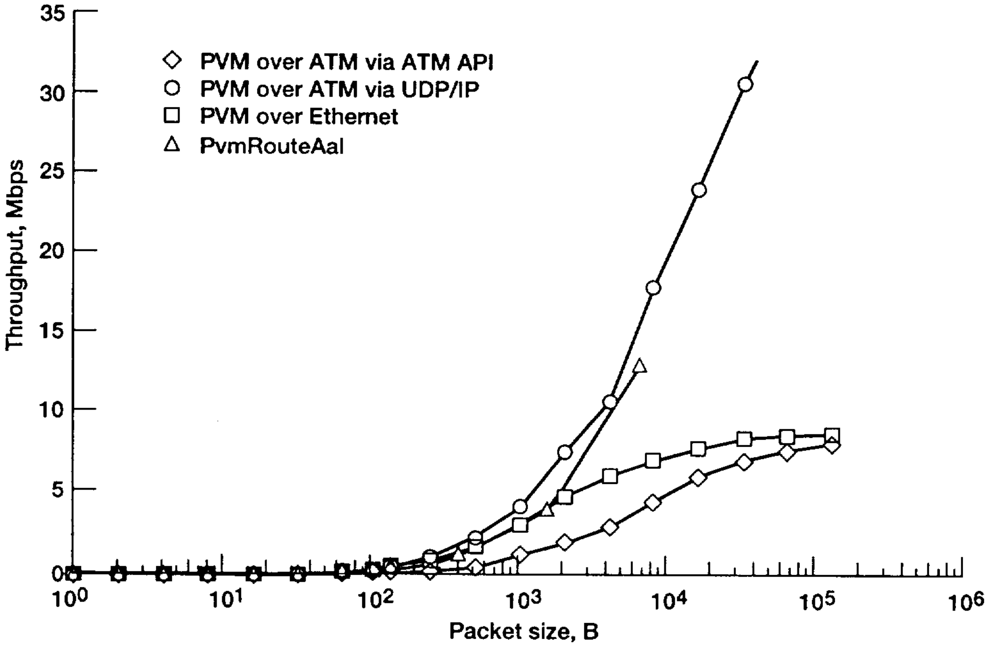

Figure 14.—Throughput versus message size with PVM showing the effect of having PVM communicate directly to the ATM API.

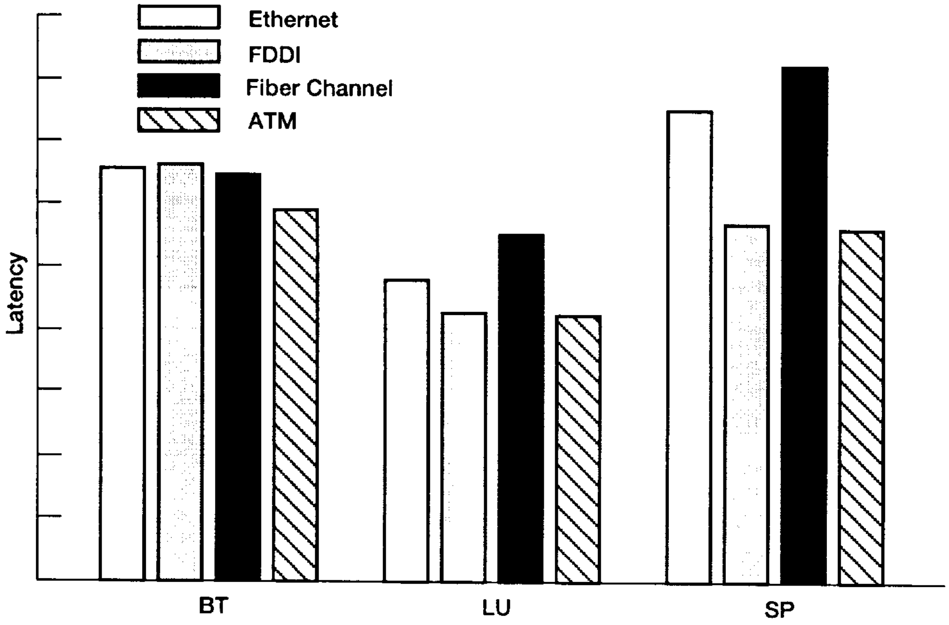

Figure 15.—Results of NAS Parallel Benchmarks (lower-upper diagonal, LU; scalar penta-diagonal, SP; and block tridiagonal, BT) with RS/6000 cluster.

Conclusions

The benefits of cluster-based computing include low-cost, flexibility, and better utilization of existing equipment. In this report we examine the effect of network technology on cluster-based computing. The main immediate bottleneck is the message-passing level and the implementation of the asynchronous transfer mode (ATM) application programmers interface (API). The performance of the various technologies is restricted by the protocols. However, performance did not degrade nearly as much as it did at the application level because of the message-passing library and the system calls when the API was invoked.

Lewis Research Center
National Aeronautics and Space Administration
Cleveland, Ohio, June 1, 1995

References

1. Geist, A., et al.: PVM 3 User's Guide and Reference Manual. Oak Ridge National Laboratory, ORNL/TM-12187, May 1994.
2. Quealy, A.; Cole, G.L.; and Blech, R.A.: Portable Programming on Parallel/Networked Computers Using the Application Portable Parallel Library. NASA TM-106238, 1993.
3. Message Passing Interface Forum: MPI: A Message-Passing Interface Standard [Online]. Available WWW: http://www.mcs.anl.gov/mpi/mpi-report/mpi-report.html Available FTP: info.mcs.anl.gov Directory: pub/ mpi File: mpi-report.ps.Z University of Tennessee, Knoxville, Tennessee, May 1994.
4. Dongarra, J., et al.: A Draft Standard for Message Passing in a Distributed Memory Environment. Proceedings of the Fifth ECMWF Workshop on the Use of Parallel Processors in Meteorology, World Scientific, Singapore, 1993, pp. 465–481.
5. Flower, J.; and Kolawa, A.: Express Is Not Just a Message Passing System: Current and Future Directions in Express. Parallel Comput., vol. 20, no. 4, Apr. 1994, pp. 597–614.
6. Minnich, R.G.; and Pryor, D.V.: Mether: Supporting the Shared Memory Model on Computing Clusters. IEEE Comput. Soc. Press, 1993, pp. 558-567.
7. Bailey, D.H., et al.: NAS Parallel Benchmark Results 3-94. Proceedings of the Scalable High Performance Computing Conference, IEEE Comput. Soc. Press, 1994, pp. 386–393.
8. Fowler, H.; and Leland, W.E.: Local Area Network Traffic Characteristics, With Implications for Broadband Network Congestion Control. IEEE. Select. Areas Commun., vol. 9, no. 7, Sept. 1991, pp. 1139–1149.
9. Lin, M., et al.: Distributed Network Computing Over Local ATM Networks. Proceedings Supercomputing '94, IEEE Comput. Soc. Press, 1994, pp. 154–163.
10. Iisaku, S.–i. and Ishikura, M.: ATM Network Architecture for Supporting the Connectionless Service. Proceedings of the IEEE INFOCOM'90, vol. 2, Jun. 1990, pp. 796–802.
11. Crocetti, P.; Gallassi, G.; and Gerla, M.: Bandwidth Advertising for MAN/ATM Connectionless Internetting. Proceedings of the IEEE INFOCOM'91, vol. 3, 1991, pp. 1145–1150.
12. Gerla, M.; Tai, T.; and Gallassi, G.: Internetting LAN's and MAN's to B–ISDN's for Connectionless Traffic Support. IEEE. Select. Areas Commun., vol. 11, no. 8, Oct. 1993, pp. 1145–1159.
13. Landegem, T.V.; and Peschi, R.: Managing a Connectionless Virtual Overlay Network on Top of an ATM Network. IEEE International Conference on Communications, 1991. pp. 988–992.
14. Aly, K.A.; and Dowd, P.W.: Parallel Computer Reconfigurability Through Optical Interconnects. Proceedings of the Twenty-first International Conference on Parallel Processing, 1992, pp. 1105–1108.
15. Aly, K.A.; and Dowd, P.W.: WDM Cluster Ring: A Low-Complexity Partitionable Reconfigurable Processor Interconnection Structure. Proceedings of the Twenty-second International Conference on Parallel Processing, 1993, pp. 1150–1153.
16. Bae, J.J.; and Suda, T.: Survey of Traffic Control Schemes and Protocols in ATM Networks. Proceedings of the IEEE, vol. 79, no. 2, Feb. 1991, pp. 170–189.
17. Eckberg, A.E.; Doshi, B.T.; and Zoccolillo, R.: Controlling Congestion in B–ISDN/ATM: Issues and Strategies. IEEE Commun. Mag., Sept. 1991, pp. 64–70.
18. Tedijanto, T.E.; and Gün, L.: Effectiveness of Dynamic Bandwidth Management Mechanisms in ATM Networks. Proceedings of the IEEE INFOCOM'93, 1993, pp. 358-367.
19. Dowd, P.W., et al.: Issues in ATM Support of High Performance Geographically Distributed Computing. Proceedings of the First International Workshop on High-Speed Network Computing, IEEE Computer Society Press, 1995, pp. 352–358.
20. Medin, M.; Srinidhi, S.M.; and Dowd, P.W.: Issues in ATM API Support for Distributed Computing. ATM Forum, 1994.
21. Jacobson, V.; Braden, R.; and Borman, D.: TCP Extensions for High Performance. RFC–1323, Internet Request for Comments No. 1323, Network Information Center, May 1992.
22. Partridge, C.: Gigabit Networking. Addison-Wesley, Reading, UK, 1994.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
September 1995

3. REPORT TYPE AND DATES COVERED
Technical Paper

4. TITLE AND SUBTITLE

Issues in ATM Support of High-Performance, Geographically
Distributed Computing

5. FUNDING NUMBERS

WU–960–30–02

6. AUTHOR(S)

Patrick W. Dowd, Saragur M. Srinidhi, Eric D. Blade, and Russell W. Claus

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–9468

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TP–3532

11. SUPPLEMENTARY NOTES

Patrick W. Dowd, State University of New York at Buffalo, Buffalo, New York 14260 (work funded by NASA Grant NAG3–1548); Saragur M. Srinidhi and Eric D. Blade, Sterling Software, 21000 Brookpark Rd., Mail Stop 142–1, Cleveland, Ohio 44135 (work funded by NASA Contract NAS3–26100); Russell W. Claus, NASA Lewis Research Center. Responsible person, Russell W. Claus, organization code 2900, (216) 433–5869.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories 32 and 33

This publication is available from the NASA Center for Aerospace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This report experimentally assesses the effect of the underlying network in a cluster-based computing environment. The assessment is quantified by application-level benchmarking, process-level communication, and network file input/output. Two testbeds were considered—one small cluster of Sun workstations and another large cluster composed of 32 high-end IBM RS/6000 platforms. The clusters had Ethernet, fiber distributed data interface (FDDI), Fibre Channel, and asynchronous transfer mode (ATM) network interface cards installed, providing the same processors and operating system for the entire suite of experiments. The primary goal of this report is to assess the suitability of an ATM-based, local-area network to support interprocess communication and remote file input/output systems for distributed computing.

14. SUBJECT TERMS

ATM; Cluster-based computing; Distributed computing

15. NUMBER OF PAGES
18

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT